2. (11 pts) Write a class named InRangeCatenate. Each instance of this class stores four fields: the lower and upper values of the range, the catenation of all the values in this range that it has seen (via the seeIt method described below), and the total number of values that it has seen (whether or not they are in the range). Its field declarations and constructor should initialize these fields appropriately (throw an IllegalArgumentException if the constructor’s first value is bigger than the second).

Its accessor, getCatenation, should return the catenation of all values seen which were in the range; its accessor, getCount, should return a count of all of values that it has seen; its mutator, seeIt, is passed a String parameter that is conditionally catenated (and always counted), changing all appropriate fields.

For example, if we write InRangeCatenate c = new InRangeCatenate("cat","mouse"); and execute the statements c.seeIt("dog"); c.seeIt("zebra"); c.seeIt("gerbil"); c.seeIt("cow"); c.seeIt("ant"); System.out.println(""+c.getCatenation()+":"+c.getCount()+"\n"); Java will print the String "dog gerbil cow:5" because "dog", "gerbil" and "cow" are all in the range specified in the constructor’s arguments (bigger than "cat" and less than "mouse" -but "zebra"and "ant" aren’t) and a total of 5 values were seen. Notice that spaces in the catenated String occur only between tokens.

Use correct syntax for defining this entire class, and all its members (constructor, methods, and fields). For full credit, write the constructor and accessors in the left column; write the mutator and fields in the right column.
2. (11 pts) Write a class named Stock. Objects from this class remember the name of a stock symbol, its current price, and the number of times that its price has changed (since construction). From the specification below, write the necessary instance variables and constructor; its field declarations and constructor should initialize these fields appropriately.

Its accessor getPrice, should return its current price (computed from its initial price and all its calls to update); its accessor getChangeCount should return the number of times its stock symbol is changed by a non-zero amount; its mutator, update, is passed a String parameter (some stock symbol) and a double (its price change). For example, if we declare Stock s = new Stock("TXN",21.5); and then call s.update("TXN",+.5); s.update("TSSC",-.5); s.update("TXN",0.); s.update("TXN",-1.0); then calling s.getPrice() returns 21.0 (21.5+.5-1.0); calling s.getChangeCount() returns 2 because the symbol TXN was changed by a non-zero amount twice. Note the call s.update("TSSC",-.5); does not update this object at all, because it has the wrong stock symbol ("TSSC" not "TXN").

Use correct syntax for defining this entire class and all its members (constructor, methods, and fields). For full credit, write the constructor and accessors in the left column; write the mutator and fields in the right column.

3. (3 pts) Objects are constructed in Java in two phases: phase 1: the new operator uses the class definition; phase 2: the constructor is called. Briefly describe what each phase does. Hint: think about what happens for a specific class like SimplifiedDiceEnsemble or Rational (or either class above) but write about the process in general.

Phase 1:

Phase 2: